METABK 59(3) 291-294 (2020)
UDC – UDK 621.74.04/658.281:536.5:519.2:519.28 = 111

A NEURAL NETWORK PREDICTION ANALYSIS OF BREAKOUT CONTINUOUS CASTING BASED ON DIFFERENTIAL EVOLUTION (DE)

Received – Primljeno: 2019-11-05
Accepted – Prihvaćeno: 2020-02-15
Original Scientific Paper – Izvorni znanstveni rad

In order to predict the breakout of continuous casting accurately and timely, a breakout prediction neural network (BPNN) model based on differential evolution algorithm was proposed. Differential evolutionary algorithm is introduced to solve the problem of fast optimization. The pretreatment of weights and thresholds improves the accuracy of breakout. The experimental analysis shows that the convergence speed of DE-BPNN breakout prediction model is faster than that of traditional neural network, and the recognition ability are significantly improved.

Keywords: continuous casting; breakout prediction; temperature; neural network; mean square error(MSE)

INTRODUCTION

Breakout is a very destructive accident in the continuous casting process. Among them, bonded breakout is the most common, accounting for more than 60 % [1]. Therefore, controlling the bonding breakout can greatly reduce the breakout rate. To solve this problem, two aspects can be considered: on the one hand, the process and equipment need to be improved; on the other hand, the establishment of early breakout prediction system should be considered. At present, thermocouple temperature measurement is the most widely used method in breakout prediction [2,3].

In view of the high complexity of breakout fault diagnosis in continuous casting, this paper proposes to establish a neural network model which integrates intelligent optimization algorithm to carry out prediction. In this model, differential evolution algorithm is introduced to optimize the input weights of the neural network, which enhances the rationality of the weights and speeds up the overall convergence speed of the algorithm. The accuracy of breakout prediction has been significantly improved[4].

PREDICTION PRINCIPLE OF BONDING BREAKOUT

The principle of bonding breakout is to use thermocouples to monitor temperature changes. The main reason is that with the movement of the billet shell and the vibration of the mould, the molten steel on the meniscus adheres to the copper plate, and breaks occur during the movement. The new high temperature molten steel on the meniscus contacts the copper plate directly. As the billet pit pulls downward and solidifies, the fracture moves downward until it leaks out at the outlet of the mould. Figure 1 shows the temperature state when bonding occurs inside the mould [5].

The solution is to give early warning and take corresponding measures before the breakout occurs, that is, during the bonding period. Thermocouples can be embedded in the copper plate of the mould. When the shell of the mould is bonded to the copper plate, there is no flux between the shell and the copper plate. After the

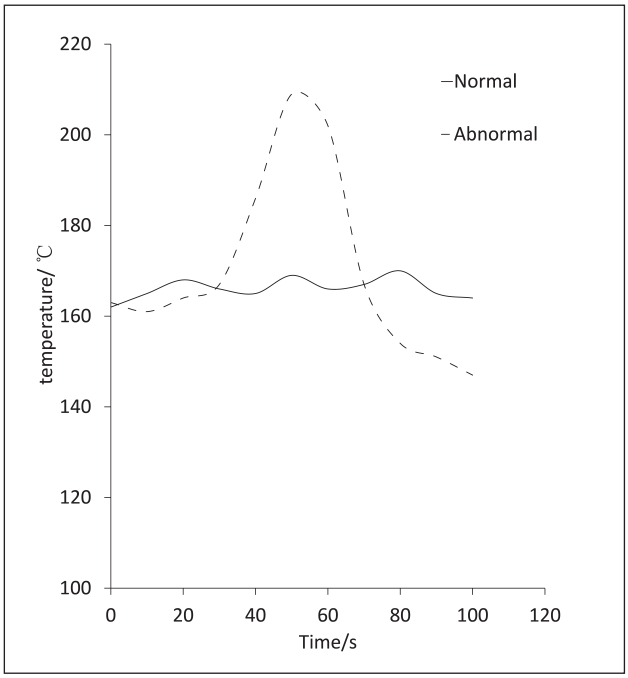

Figure 1 Temperature characteristic change of bonded breakout

Y. R. Li. e-mail: lyr7879@163.com, Institute of Applied Technology, University of Science and Technology Liaoning, China; C. N. Zhang, School of Software, University of Science and Technology Liaoning, China

shell is broken, it contacts directly with the copper plate, which makes the temperature of the lower exhaust thermocouple rise rapidly. When the temperature difference reaches a certain limit, it can give a warning.

BREAKOUT MODEL

Analysis of differential evolution algorithms

The basic idea of Differential evolution (DE) is similar to that of Genetic algorithms (GA). It uses mutation operation to generate new individuals, then carries out crossover and selection operation, and searches for the global optimal solution through continuous iterative evolution. It implements mutation operation through difference strategy, which is different from genetic algorithm, and uses group characteristics to enhance the optimization ability of the algorithm[6]. In the optimization problem, taking the solution of the non-linear minimization problem as an example, the basic mathematics is as follows:

$$\begin{cases} \min f(x_1, x_2, \text{L}\ x_Q) \\ x_j^L \le x_j \le x_j^U, j = 1,2,\text{L}\ ,Q \end{cases} \quad (1)$$

In the formula, Q is the solution space dimension, U, L are the upper and lower limits of component x_j respectively.

The basic steps of DE algorithm are as follows:

Population initialization

Setting the population size is N_p, individual i of the k generation can be expressed as $\{x_{j,i} \mid x_{j,i}^L \le x_{j,i}^L x_{j,i} \le x_{j,i}^U, i = 1,2,\text{L}\ ,N_p;\ j = 1,2,\text{L},\ Q\}$, each generation population in the algorithm consists of N_p vectors with dimension Q, and the initial generation population $x_{j,i}(0)$ can be expressed as:

$$x_{j,i}(0) = x_{j,i}^L + rand(0,1) \cdot (x_{j,i}^U - x_{j,i}^L) \quad (2)$$

In the formula, *rand* (0,1) is a random number, $\in$ (0,1), uniform distribution.

Mutation operation

For each target individual $x_{j,i}$ in the population, DE usually uses differential strategy to implement mutation operation, that is, two individuals are randomly selected in the population and the individual to be mutated are fused with vectors to produce a new mutation individual $V_{j,i}(k + 1)$.

$$V_{j,i}(k+1) = x_{r,1}(k) + F \cdot (x_{r2}(k) - x_{r,3}(k)) \quad (3)$$

In the formula, $V_{j,i}(k + 1)$ is a new variant individual; F is the scaling factor, $\in$ (0,1), r_1, r_2, $r_3 \in [1, N_p]$ is used to control the vector difference of randomly selected individuals, which is also different from the individual to be mutated, at the same time, $r_1 \ne r_2 \ne r_3 \ne j$.

In the iteration of the algorithm, both randomly selected individuals and newly generated individuals must be effective, namely, $\{V_{j,i}(k+1) | x_{j,i}^L \le V_{j,i}(k+1) \le x_{j,i}^U\}$.

Cross operation

Individual $V'_{j,i}(k + 1)$ is obtained by the cross transformation of the individual, and the corresponding $V'_{j,i}(k + 1) = \{V'_{1,i}(k + 1), V'_{2,i}(k + 1), \text{L}, V'_{Q,i}(k + 1)\}$ is obtained. For the intermediate crossover transformation, the random strategy is used to make the individual under test contain at least one vector generated by $x_{j,i}$. Then the crossover must follow the following formulas:

$$V'_{j,i}(k+1) = \begin{cases} V_{j,i}(k+1) & rand(j) \le CR \text{ or } j \in [1,Q] \\ x_{j,i}(k) \end{cases} \quad (4)$$

In the formula, j is a limited random number to ensure that the individual to be tested does not completely depart from the target individual $x_{j,i}$, $CR \in [0,1]$, is called crossover probability. It is used to adjust the difference between the newly generated individual and the original individual. It is a random number.

Selection operation

Comparing the fitness values of the individual to be tested and the target individual, the smaller one is retained to become the next generation of new individuals. The greedy algorithm is used to determine the new individuals:

$$x_{j,i}(k+1) = \begin{cases} V'_{j,i}(k+1) & f(V'_{j,i}(k+1)) \le f(x_{j,i}(k)) \\ x_{j,i}(k) & f(V'_{j,i}(k+1)) > f(x_{j,i}(k)) \end{cases} \quad (5)$$

BPNN optimization

BP neural network has the characteristics of reverse transmission, self-learning and self-adaptation, including input layer, hidden layer and output layer. The corresponding topological structure is shown in Figure 2.

As can be seen from the above figure, the three-layer neural network can realize the mapping from the input layer to the output layer, that is, m-dimension to n-dimension. The transfer function at both ends is Tansig by default, the learning function of weight and threshold is sigmoid by default, the number of neurons in the hidden layer is l, the weight associated with the input layer and the hidden layer is set to w_{ij}, the weight associated with the output layer is v_{jk}, the j neuron in the hidden layer can be expressed as λ_j, and the threshold of k neuron in the output layer can be expressed as η_k. For $\overline{y}_k = \{\overline{y}_1, \overline{y}_2, \text{L}\ , \overline{y}_l\}$, output can be obtained:

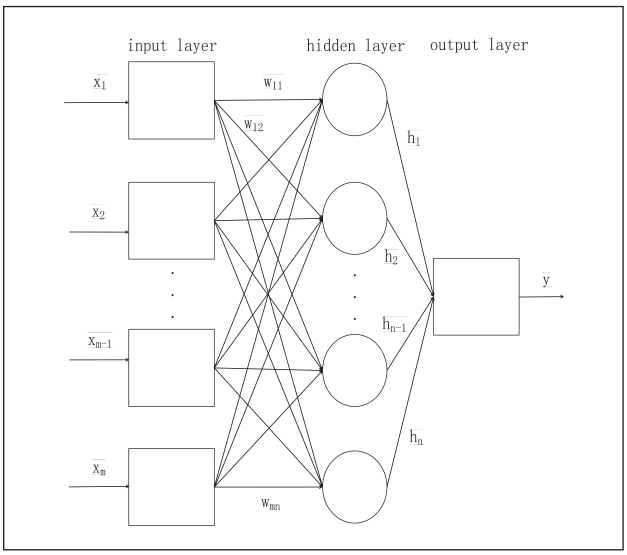

Figure 2 Topology of BPNN

$$\overline{y}_k = \sum_{j=1}^{l} v_{jk}\lambda_j - \eta_j \tag{6}$$

The number of nodes in input layer and hidden layer should be satisfied: $n \leq 2m + 1$. In the formula, m is the number of nodes in the input layer and n is the number of nodes in the hidden layer. Then the error function is:

$$E_i = \frac{\sum_{i=1}^{N}(y_i - \overline{y}_i)^2}{2} \tag{7}$$

In the formula, y_i, $\overline{y}_i$ are expected output values and actual output values respectively, N is sample number. In this paper, the number of hidden layers is set to 1. The number of neurons is very important. The excessive number of neurons and the increasing amount of network computation will easily lead to over-fitting. If the number of neurons is too small, the network calculation performance will be insufficient and can't be fitted. Although the weights and thresholds are the minimum values obtained randomly, they still have a great impact on the output results, which easily leads to slow convergence speed and local optimum problems.

Optimization of breakout model

Aiming at the problems of BPNN in breakout analysis, this paper uses DE algorithm to solve the weights and thresholds of BPNN, applies the optimal solution to BPNN, and carries out follow-up training to compensate for the errors. Set the fitness function as follows:

$$f(x_i) = \begin{cases} 1/(E_i + 1) & E_i > 0 \\ 1 & E_i = 0 \end{cases} \tag{8}$$

In the formula, $f(x_i)$ is the fitness function of DE algorithm and E_i is the value of error function.

The specific operation steps are as follows:

(1) Initialize the parameters of DE algorithm and BPNN, including the population size N_p, the number of iterations C, the number of nodes (m, n, l) in the input layer, the hidden layer and the output layer, the weight w of the input layer and the hidden layer, the weight v of the hidden layer and the output layer, the threshold η of the hidden layer and the output layer, and the input and output data are normalized.

(2) The fitness value of each solution in the evolution is calculated according to formula (8) to obtain the optimal weight and threshold. The ideal case of fitness is $f(x_i) = 1$.

(3) In each iteration of evolution, a new feasible solution $V'_{j,i}(k + 1)$ is obtained, which is compared with the solution of the previous generation, and the replacement operation is implemented by using formula (5).

(4) Constantly evolved, when the fitness value meets the requirements, the optimal solution is obtained, that is, the calculation of the ownership value and the threshold value is completed, and it is trained as the basis for constructing the BPNN.

EXPERIMENTAL ANALYSIS

Because of the complexity of the working condition data, it is difficult to avoid the singular sample data. Normalization is needed to limit the singular sample data to [0,1]. The reference formula is as follows:

$$x'_i = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}} \tag{9}$$

In the formula, $x = \{x_1, x_2, \mathrm{L}, x_n\}$ is the original sample data, n is a dimension, $x_{\max}$, $x_{\min}$ are the maximum and minimum values of samples respectively. The population size of DE algorithm is 80, the crossover probability is 0,3, the scaling factor is 0,5 and the number of iterations is 100. The number of iterations of BPNN is 200, and the target error is 10^{-4}. In practice, 180 groups of field data were selected as training samples, and BPNN and DE-BPNN were compared respectively. The fitness values of the weights calculated by DE algorithm were as follows figure 3:

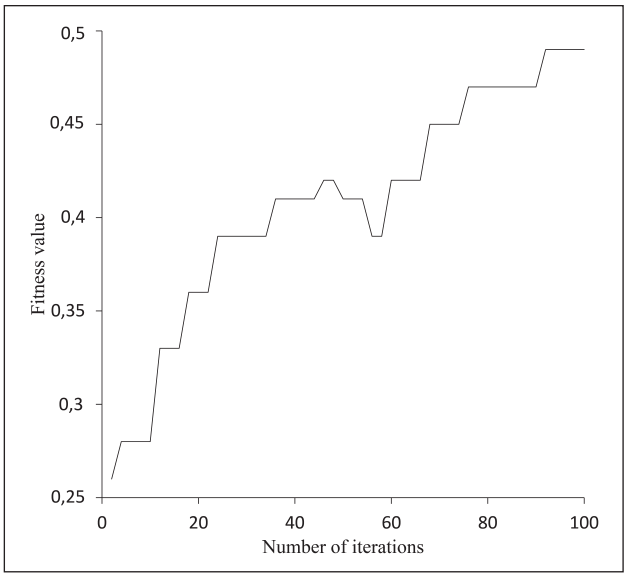

Figure 3 Algorithmic fitness value

The above figure shows that with the iteration, the fitness value rises rapidly. When the solution is cross-selected nearly 60 times, the solution is replaced by the new optimal solution, the fitness changes slightly, and then returns to the ascending orbit. This also proves that the DE algorithm is effective in preprocessing the weights and thresholds.

The analysis of mean square error of two algorithms, BPNN and DE-BPNN, is as follows figure 4:

As can be seen from the above figure, the error of DE-BPNN is much lower than that of BPNN in general. The diagnosis of breakout is better, and the error curve of the improved neural network decreases faster, converges faster and has stronger stability.

CONCLUSIONS

In this paper, a new breakout prediction model is constructed by introducing DE algorithm, which com-

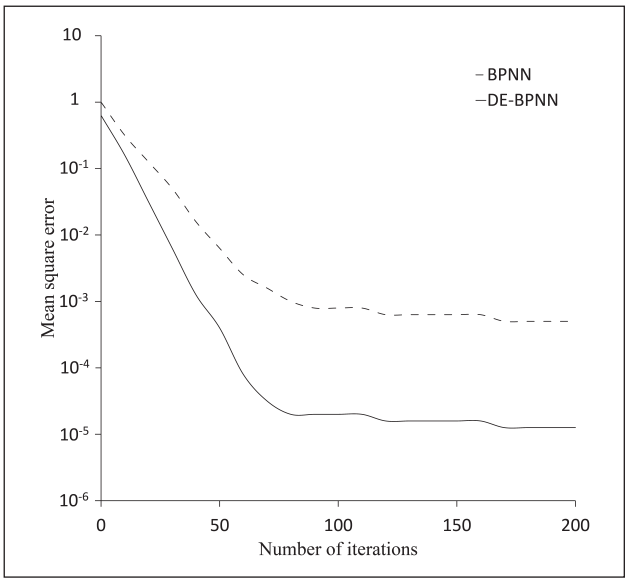

Figure 4 Mean square error curve

bines the global optimization ability of DE algorithm with neural network, preprocesses the weights and thresholds, and overcomes the slow convergence speed of traditional neural network and easily falls into local optimum. In the breakout prediction, the identification and accuracy of the typical bond breakout prediction have been effectively improved.

Acknowledgments

This work was supported by the science and technology support program (2016HZPY09) from Liaoning Education Department.

REFERENCE

[1] Ansari Md Obaidullah, Ghose J., Kumar, R.. Neural network based breakout predicting system for all four strands of caster in a continuous casting shop—A case study[J]. Advances in Intelligent Systems and Computing 509(2017), 89-99.

[2] Ito Y,Nabeshima S, Miki Y, Kubota J, Matsuoka K. Evaluation of Solidified Shell Thickness by Thermocouple in Mold[J]. Tetsu To Hagane-Journal Of The Iron And Steel Institute Of Japan 104(2018)8, 36-43.

[3] Ray K, Basak I. Local heat flux profiles and interfacial thermal resistance in steel continuous casting[J]. Journal Of Materials Processing Technology 255(2018), 605-610.

[4] Mohammed MS, Vural RA.NSGA-II plus FEM Based Loss Optimization of Three-Phase Transformer[J]. Ieee Transactions On Industrial Electronics 66(2019)9, 7417-7425.

[5] He F, Zhang LY. Mold breakout prediction in slab continuous casting based on combined method of GA-BP neural network and logic rules[J]. International Journal Of Advanced Manufacturing Technology 95(2018)9-12, 4081-4089.

[6] Khaparde, A.R., Raghuwanshi, M.M, Liqun and Malik, L.G.. A new distributed differential evolution algorithm[C]. International Conference on Computing, Communication and Automation (2015), Greater Noida, India, 558-562.

Note: The responsible for English is Zhang Yue Ru, Liaoning, China